Statistical Optimization of Non-Negative Matrix Factorization

Anoop Korattikara, Levi Boyles, Max Welling
{akoratti, lboyles, welling}@ics.uci.edu
Department of Computer Science
University of California, Irvine

Jingu Kim, Haesun Park
{jingu,hpark}@cc.gatech.edu
College of Computing
Georgia Institute of Technology

Abstract

Non-Negative Matrix Factorization (NMF) is a dimensionality reduction method that has been shown to be very useful for a variety of tasks in machine learning and data mining. One of the fastest algorithms for NMF is the Block Principal Pivoting method (BPP) of [7], which follows a block coordinate descent approach. The optimization in each iteration involves solving a large number of expensive least squares problems. Taking the view that the design matrix was generated by a stochastic process, and using the asymptotic normality of the least squares estimator, we propose a method for improving the performance of the BPP method. Our method starts with a small subset of the columns and rows of the original matrix and uses frequentist hypothesis tests to adaptively increase the size of the problem. This achieves two objectives: 1) during the initial phase of the algorithm we solve far fewer, much smaller sized least squares problems and 2) all hypothesis tests failing while using all the data represents a principled, automatic stopping criterion. Experiments on three real world datasets show that our algorithm significantly improves the performance of the original BPP algorithm.

1 Introduction

Non-Negative Matrix Factorization (NMF) is a popular dimensionality reduction technique that has many useful applications in machine learning. NMF is typically applied to high dimensional data where each element is non-negative and it provides a low rank approximation formed by factors whose elements are themselves non-negative. Unlike other dimensionality reduction techniques, the non-negativity constraints in NMF typically lead to a "sum of parts" decomposition of objects. NMF has been applied successfully to a variety of tasks in fields such as computer vision, text mining, spectral analysis and bioinformatics.

NMF can be formulated mathematically as follows. Given an input matrix $\mathbf{A} \in \mathbb{R}^{p \times q}$ where each element is nonnegative and an integer $k < min\{p, q\}$, NMF aims to find two factors $\mathbf{W} \in \mathbb{R}^{p \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times q}$ with nonnegative elements such that $\mathbf{A} \approx \mathbf{WH}$. The factors $\mathbf{W}$ and $\mathbf{H}$ are commonly found by solving the following non-convex optimization problem:

$$\min_{\mathbf{W},\mathbf{H}} f(\mathbf{W}, \mathbf{H}) = \frac{1}{2}\|\mathbf{A} - \mathbf{WH}\|_F^2 \tag{1}$$

$$\text{subject to } \forall ij, \mathbf{W}_{ij}, \mathbf{H}_{ij} \geq 0$$

Recent studies [11, 6, 7] have demonstrated that methods based on the Alternating Nonnegative Least Squares (ANLS) framework are computationally efficient. The ANLS framework is a simple alternating minimization scheme that iteratively optimizes each of the factors $\mathbf{W}$ and $\mathbf{H}$ keeping the other fixed. The ANLS framework has nice convergence properties provided by a block coordinate descent argument [1]. We spell out the general form of ANLS in Algorithm 1.

Although the original NMF problem in Eqn. (1) is non-convex, the subproblems in Eqns (2) are convex and are instances of the Non-negativity Constrained Least Squares (NNLS) problem. One of the most efficient algorithms for solving NNLS problems is the Block Principal Pivoting (BPP) method [7], which we use as the backbone for our optimization procedure.

The BPP algorithm involves computing $\mathbf{W}^T\mathbf{W}$, $\mathbf{W}^T\mathbf{A}$, $\mathbf{HH}^T$ and $\mathbf{HA}^T$ for solving the least squares problems, which can be very expensive when either p or q is large. We reinterpret the least squares problems embedded in BPP as a statistical estimation problem,

Appearing in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15. Copyright 2011 by the authors.

Algorithm 1 ANLS Framework for NMF

```
1. Initialize $\mathbf{W} \in \mathbb{R}^{p \times k}$ with nonnegative elements
2. Repeat solving the following problems until convergence:
       $\min_{\mathbf{H} \geq 0} \|\mathbf{W}\mathbf{H} - \mathbf{A}\|_F^2$ where $\mathbf{W}$ is fixed   (2a)
       $\min_{\mathbf{W} \geq 0} \|\mathbf{H}^T\mathbf{W}^T - \mathbf{A}^T\|_F^2$ where $\mathbf{H}$ is fixed   (2b)
3. The columns of $\mathbf{W}$ are normalized to unit $L_2$ norm and the rows of $\mathbf{H}$ are scaled accordingly.
```

acknowledging the fact that the data matrix $\mathbf{A}$ was generated through a stochastic process. In essence, we study the expected fluctuations under resampling $\mathbf{A}$ from its generating process. Under such resampling, the least squares estimator is known to be asymptotically normal if certain conditions are satisfied [17].

Since $\mathbf{W}$ and $\mathbf{H}$ are far from their target values in the early iterations, we do not need all the available data to compute the updates as in Eqns. (2). Instead, a few samples can provide a rough estimate of the update direction in parameter space. The proposed algorithm sub-samples the rows of $\mathbf{W}$ and $\mathbf{A}$ for computing $\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{A}$, and sub-samples the columns of $\mathbf{H}$ and $\mathbf{A}$ for computing $\mathbf{H}\mathbf{H}^T$ and $\mathbf{H}\mathbf{A}^T$. The asymptotic normality of the least squares estimator allows us to formulate statistical hypothesis tests to determine whether we have enough evidence to carry out the updates to $\mathbf{W}$ and $\mathbf{H}$ using the current sample size, and if not, increase the sample size to reduce the uncertainty in our updates. The procedure not only reduces the number of operations to compute the required matrix products but also reduces the number of least squares problems we are required to solve.

Another advantage is that this statistical view suggests a natural stopping criterion for the optimization procedure. Most optimization algorithms check for convergence using a tolerance parameter that is more related to machine precision than to the precision determined by the statistical properties of the data. We argue that a more principled termination criterion would be to stop optimizing when the proposed updates fail the hypothesis tests after using all the available data.

Finally, we would like to stress that our method is not restricted to NMF optimization, but in fact represents a general philosophy that can be applied to a variety of iterative optimization procedures as long as the parameter updates involve large sums of random variables. The more general point we wish to convey is that statistical estimation is not merely optimization of a loss function. Instead, the stochastic nature of the data induces an intrinsic *statistically determined* length scale. Optimizing beyond this is not only computationally costly but may even lead to overfitting. Moreover, the required precision for parameter updates also depends on how far learning has progressed: one needs more data close to the optimum than during the initial phase of learning. These ideas seem applicable to a very broad range of statistical estimation problems, and as our experiments show, they lead to significant improvements in performance for NMF problems, where we were able to speed up the current state-of-the-art algorithm (BPP) by a factor of 4 or 5 on large datasets. This is very notable, considering the fact that BPP already achieves a very high speed-up relative to other popular NMF algorithms [7].

The rest of the paper is organized as follows. We review related work in Section 2 and briefly describe the Block Principal Pivoting algorithm in Section 3. Section 4 describes our method for optimizing NMF by sub-sampling the data. We present experimental results in Section 5 and conclude in Section 6.

2 Related Work

A major computational challenge in NMF is to solve the NNLS problems. A number of methods have been proposed for this, starting with the classical algorithm by Lawson and Hanson [9]. Lin [11] developed a gradient descent based algorithm with a projection on to the nonnegative orthant. D. Kim et al.[5] suggested a quasi-Newton method with projections for faster convergence. H. Kim and Park[6] studied an improved active-set algorithm, and J. Kim and Park[7] proposed the efficient BPP method, which we optimize.

Our method is also related, although not identical, to Stochastic Approximation (SA) [15, 8, 16], a class of iterative algorithms that can be used to minimize a loss function using only noisy gradients. In SA, the updates at each step are multiplied with a gain value, that decreases over time to ensure convergence. Thus, a key difference is that in SA, the *effect of uncertainty* is reduced by using smaller and smaller gain values, whereas, we systematically reduce the *uncertainty in the updates* by using larger and larger batches. The idea of increasing batch size to ensure convergence has been explored in [10, 14, 3], and using univariate hypothesis tests to determine the optimal batch size has been studied in [3].

In SA, although it is easy to choose a gain sequence that ensures convergence, it requires a lot of tuning to find the one that gives the best performance. In contrast, our method has a single interpretable parameter (the significance level for hypothesis tests) and has a statistically principled stopping criterion. Additionally, unlike our NMF algorithm where the non-negativity constraints are handled by the BPP method, SA based methods have to deal with this by projecting an unconstrained version of the solution onto the non-negative orthant. An example of an SA approach to NMF is [12], where the authors propose an online method for dictionary learning and study NMF as a special case. However, they focus on learning only one of the factors (the matrix $\mathbf{W}$ according to our terminology), whereas our method accelerates learning of both the factors.

Our ideas also have a close connection to the observations in [2]. Usually in learning problems, one is interested in minimizing an *expected* loss function with respect to some true underlying distribution of the data. However, in practice, one works with an *empirical* loss function defined with respect to a finite dataset, that is only an approximation of the expected loss. Therefore, it is wasteful to expend too much computational time in minimizing the empirical loss to very high accuracy.

3 Block Principal Pivoting Algorithm for Solving NNLS problems

We will now briefly review the Block Principal Pivoting algorithm for solving the NNLS problems (2a) and (2b). For simplicity, let us consider an NNLS problem with only a single right-hand side vector:

$$\min_{\mathbf{x}\geq 0} \|\mathbf{C}\mathbf{x}-\mathbf{b}\|_2^2 \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{n\times d}$, $\mathbf{x} \in \mathbb{R}^{d\times 1}$ and $\mathbf{b} \in \mathbb{R}^{n\times 1}$. The subproblems in Eqns. (2) can be decomposed into several instances of (3). Thus, an algorithm for (3) is a basic building block for an algorithm for Eqns (2).

If $\mathbf{C}$ has full rank, $\mathbf{C}^T\mathbf{C}$ is positive definite and the problem in Eqn. (3) is strictly convex. Then, a solution $\mathbf{x}$ that satisfies the following Karush-Kuhn-Tucker (KKT) conditions is the optimal solution to Eqn. (3):

$$\mathbf{y} = \mathbf{C}^T\mathbf{C}\mathbf{x} - \mathbf{C}^T\mathbf{b} \tag{4a}$$
$$\mathbf{y} \geq \mathbf{0} \tag{4b}$$
$$\mathbf{x} \geq \mathbf{0} \tag{4c}$$
$$\mathbf{x}_i\mathbf{y}_i = \mathbf{0}, i = 1, ..., d \tag{4d}$$

To solve this, the index set $1, ..., d$ is first divided into two sets F and G where $F \cup G = 1, ..., d$ and $F \cap G = \phi$. Let $\mathbf{x}_F$, $\mathbf{x}_G$, $\mathbf{y}_F$ and $\mathbf{y}_G$ denote the subsets of variables with corresponding indices, and let $\mathbf{C}_F$ and $\mathbf{C}_G$ denote the sub-matrices of $\mathbf{C}$ with the corresponding column indices. Initially, $\mathbf{x}_G$ and $\mathbf{y}_F$ are set to $\mathbf{0}$. By construction, $\mathbf{x} = (\mathbf{x}_F, \mathbf{x}_G)$ and $\mathbf{y} = (\mathbf{y}_F, \mathbf{y}_G)$ satisfy Eqn. (4d). We can solve for $\mathbf{x}_F$ and $\mathbf{y}_G$ using Eqn. (4a) :

$$\mathbf{x}_F = (\mathbf{C}_F^T\mathbf{C}_F)^{-1}\mathbf{C}_F^T\mathbf{b} \tag{5a}$$
$$\mathbf{y}_G = \mathbf{C}_G^T\mathbf{C}_F\mathbf{x}_F - \mathbf{C}_G^T\mathbf{b} \tag{5b}$$

If, $\mathbf{x}_F \geq \mathbf{0}$ and $\mathbf{y}_G \geq \mathbf{0}$, $\mathbf{x} = (\mathbf{x}_F, \mathbf{0})$ is a solution. Otherwise, the index sets F and G are updated by exchanging variables for which Eqn. (4b) or Eqn. (4c) does not hold, and the process is repeated till a feasible solution for $\mathbf{x}$ is found. Further details of this method, including its efficient extension to the multiple right hand side case is described in [7].

4 Statistical Optimization

We saw that in each iteration of the ANLS algorithm, we update the $\mathbf{x}$'s by solving least squares problems as in Eqn. (5a). Since n, the number of rows in $\mathbf{C}$ and components in $\mathbf{b}$, can be very large, computing $\mathbf{C}_F^T\mathbf{C}_F$ and $\mathbf{C}_F^T\mathbf{b}$ to solve Eqn. (5a) can be very expensive.

The crucial step to cut down on computation is to view the rows of $\mathbf{C}_F$ as samples from a stochastic process. To simplify notation, we will suppress the F index in $\mathbf{C}_F$ and $\mathbf{x}_F$ and write $\mathbf{C}$ and $\mathbf{x}$ from now on, unless we need to make the distinction explicit. Let us denote the i^{th} row of $\mathbf{C}$ as $\mathbf{c}^{(i)T}$ and the i^{th} component of $\mathbf{b}$ as $b^{(i)}$, and interpret them as instances of random variables $\mathbf{c}$ and b respectively. Interpreting $\mathbf{x}$ as a model parameter, we may introduce the following generative stochastic process $b = \mathbf{c}^T\mathbf{x} + \epsilon$, where ϵ is an error term. Importantly, the maximum likelihood estimator of $\mathbf{x}$ is exactly given by least squares solutions of the form Eqns. (5a). Our strategy will be to subsample the data instances $\{\mathbf{c}^{(i)}, b^{(i)}\}$ used for estimating $\mathbf{x}$, or equivalently the rows of $\mathbf{C}$ and $\mathbf{b}$.

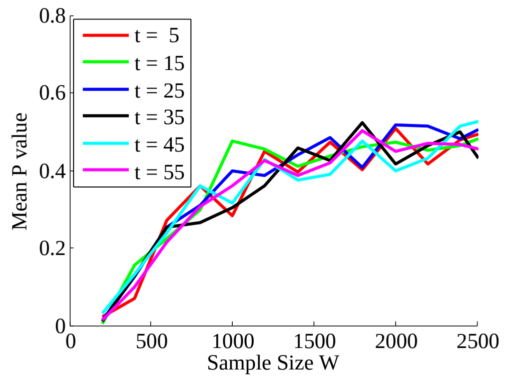
If we further assume that the following conditions hold,

1. No multicollinearity: $\mathbf{Q}_{cc} = \mathbb{E}[\mathbf{c}\mathbf{c}^T]$ is positive definite
2. Exogeneity: $\mathbb{E}[\epsilon|\mathbf{c}] = 0$
3. Homoscedasticity: $Var[\epsilon|\mathbf{c}] = \sigma^2$

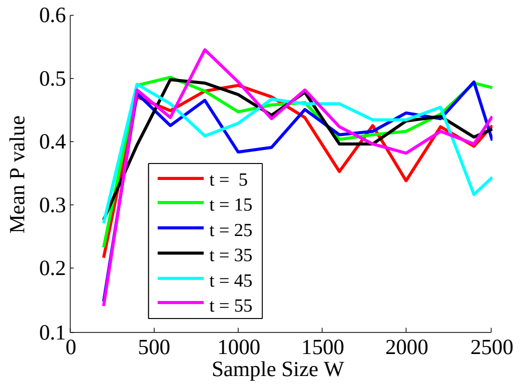
then the least squares estimator for $\mathbf{x}$ is known to be asymptotically normal [17]:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \text{ where } \boldsymbol{\Sigma} = \frac{\mathbf{Q}_{cc}^{-1}\sigma^2}{n} \tag{6}$$

The assumptions above and the assumption that $\{\mathbf{c}^{(i)}, b^{(i)}\}$ represent i.i.d. samples from some distribution may obviously be violated for real datasets. However, the property that we need in practice is that the



(a)



(b)

Figure 1: Normality testing: Results of Mardia's skewness (Fig. 1a) and kurtosis (Fig. 1b) test for multinormality of the least squares estimator while solving for $\mathbf{H}$, keeping $\mathbf{W}$ fixed on the AT&T faces dataset with k = 16. Mean p-values of the tests for various sample sizes of $\mathbf{W}$ at different iterations (t) are shown.

estimator for $\mathbf{x}$ approximately follows a normal distribution. We therefore tested for this property using Mardia's multivariate skewness and kurtosis test [13] at different iterations of the algorithm (to be described below). We drew many samples of size n, solved the corresponding least squares problems and measured the p-value (see Figure 1). There is clearly no evidence for rejecting the null hypothesis that the least squares estimator is normal, when $n > 500$. It should be mentioned that the normality assumption does not hold for very sparse matrices and the performance of our algorithm will be unpredictable in such cases.

Note from Eqn. (6) that as the number of samples, n, increases, the uncertainty in $\mathbf{x}$ decreases. In the early iterations of the ANLS algorithm, we are far from the target value of $\mathbf{x}$ and we do not really need the precision from using all the samples $\{\mathbf{c}^{(i)}, b^{(i)}\}$ to update our estimate of $\mathbf{x}$. Instead, a few samples can tell us the general direction in which to move. As we progress further towards convergence, we can use more samples to reduce the uncertainty in the solution and make the updates more precise. This sampling procedure comes with two main computational advantages. First, we need many fewer operations to compute $\mathbf{C}^T\mathbf{C}$ and $\mathbf{C}^T\mathbf{b}$ in the early iterations. Second, the number of least squares problems to be solved in each iteration is reduced. For example, if in iteration $t+1$, we are going to use only n_{t+1} rows of $\mathbf{W}$ to solve for the columns of $\mathbf{H}$, then in iteration t, we only had to solve for these n_{t+1} rows of $\mathbf{W}$. As learning proceeds, more and more rows and columns get involved. We now discuss how to estimate the number of rows and columns that are needed at at any stage of the algorithm.

Our approach is to make sure that the direction of the update, is at least within 90 degrees of the true update direction with high probability. We can check this using statistical hypothesis tests and if we fail the hypothesis test with our current sample size, we can query more samples to reduce the uncertainty in the update direction. To formulate any hypothesis tests about the update direction, we should first determine the distribution of the proposed update under different samplings of the data. Let us denote the estimate of $\mathbf{x}$ determined in the t^{th} iteration by the random variable $\mathbf{x}_t$ and an instantiation of this random variable by $\hat{\mathbf{x}}_t$. Since $\mathbf{x}_t$ is distributed normally as in (6), we can approximate this distribution as $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ where:

$$\boldsymbol{\mu}_t \approx \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{n_t} (\mathbf{c}^{(i)T}\mathbf{x} - b^{(i)})^2 \tag{7a}$$

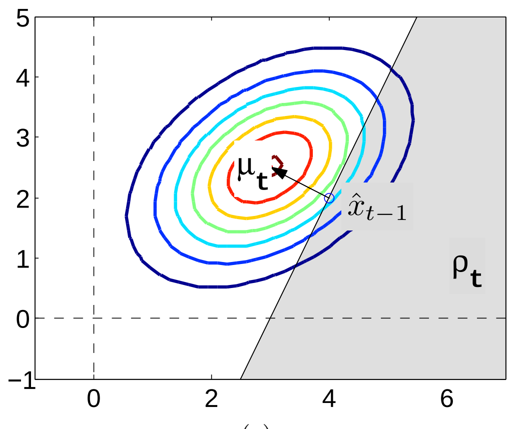
$$\mathbf{Q}_{cc} \approx \frac{1}{n_t - 1} \sum_{i=1}^{n_t} \mathbf{c}^{(i)}\mathbf{c}^{(i)T} \tag{7b}$$

$$\sigma^2 \approx \frac{1}{n_t - 1} \sum_{i=1}^{n_t} (\epsilon^{(i)})^2 \tag{7c}$$

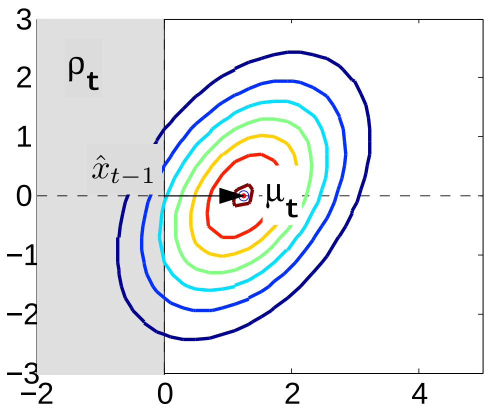
$$\boldsymbol{\Sigma}_t \approx \frac{\mathbf{Q}_{cc}^{-1}\sigma^2}{n_t} \tag{7d}$$

We are now in a position to formulate hypothesis tests to determine whether we have enough statistical evidence to update our current estimate, $\hat{\mathbf{x}}_{t-1}$, of $\mathbf{x}$ with $\boldsymbol{\mu}_t$, the mean of the distribution of $\mathbf{x}_t$. Consider a hyperplane $\mathcal{P}$ passing through $\hat{\mathbf{x}}_{t-1}$, orthogonal to the vector $\boldsymbol{\ell} = \boldsymbol{\mu}_t - \hat{\mathbf{x}}_{t-1}$. $\mathcal{P}$ partitions $\mathbb{R}^d$ into two half-spaces as shown in Figure 2a. The probability ρ_t of lying in the half-space that does not contain $\boldsymbol{\mu}_t$ is the probability that the update $\boldsymbol{\mu}_t$ is in the wrong direction, and we want this value to be lower than some threshold Δ in order to not reject the null hypothesis that we are moving in the correct direction.

To compute ρ_t, consider a transformation of the co-



(a)



(b)

Figure 2: Hypothesis testing: In 2a, $\mathbf{x}$'s old value of $\hat{\mathbf{x}}_{t-1}$ is about to be replaced with $\boldsymbol{\mu}_t$. The shaded region, ρ_t represents the probability that the proposed update direction is not within 90 degrees of the true direction. 2b shows how transforming to an alternate coordinate system can help compute ρ_t efficiently.

ordinate system so that $\hat{\mathbf{x}}_{t-1}$ is at the origin and $\boldsymbol{\ell}$ is aligned with the first coordinate axis. In this new co-ordinate system $\mathbf{x}_t$ is distributed as:

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}') \text{ where } \boldsymbol{\mu}' = (\|\boldsymbol{\ell}\|, \mathbf{0}) \text{ and } \boldsymbol{\Sigma}' = \mathbf{S}\boldsymbol{\Sigma}\mathbf{S}^{-1} \tag{8}$$

Here $\mathbf{S}$ is the rotation matrix that aligns $\boldsymbol{\ell}$ with the first coordinate axis (see Appendix A). Since, in this new co-ordinate system, $\boldsymbol{\ell}$ lies along the first coordinate axis, $\mathcal{P}$ is just the hyperplane containing all other coordinate axes. $\mathcal{P}$ divides $\mathbb{R}^d$ into two half spaces, $\mathbb{R}^d_+$ containing the positive part of the first coordinate axis and $\mathbb{R}^d_-$ containing the negative part, as in Figure 2b. Now, ρ_t is just the probability of falling in $\mathbb{R}^d_-$ and can be computed from the marginal distribution of $\mathbf{x}_{t,1}$, the first component of $\mathbf{x}_t$. This marginal distribution is just $\mathcal{N}(\boldsymbol{\mu}'_1, \sqrt{\boldsymbol{\Sigma}'_{11}})$ and we calculate $\rho_t = \Phi(0)$, where $\Phi(.)$ is the CDF of $\mathbf{x}_{t,1}$.

Algorithm 2 BPP Algorithm with Sub-Sampling

```
1. Initialize $S_W$ and $S_H$, the initial sample sizes for $\mathbf{W}$ and $\mathbf{H}$, to a fraction of $p$ and $q$.
2. Initialize $\mathbf{W} \in \mathbb{R}^{p \times k}$ with nonnegative elements.
3. Randomly permute the rows and columns of $\mathbf{A}$, so that picking the first $n$ rows or columns of $\mathbf{A}$ is equivalent to drawing $n$ random samples.
4. repeat solving subproblems (a) and (b) until convergence:
   (a) $\min_{\mathbf{H} \geq 0} \|\mathbf{WH} - \mathbf{A}\|_F^2$ where $\mathbf{W}$ is fixed:
       // determine required sample size $S_W$
       i. while $S_W < p$
          • Solve for $J$ randomly chosen columns of $\mathbf{H}$ by the BPP method but using only $S_W$ rows of $\mathbf{W}$ and $\mathbf{A}$.
          • Conduct $J$ hypothesis tests to see if the update directions are correct.
          • if any of the hypothesis tests fail
            – $S_W \leftarrow min(p, 2S_W)$
            – Since we increased $S_W$, solve for these new samples of $\mathbf{W}$ using $\mathbf{H}$ from the previous iteration.
            else
            – break
       ii. Solve for the first $S_H$ columns of $\mathbf{H}$ using $S_W$ rows of $\mathbf{W}$ and $\mathbf{A}$
   (b) $\min_{\mathbf{W} \geq 0} \|\mathbf{H}^T\mathbf{W}^T - \mathbf{A}^T\|_F^2$ where $\mathbf{H}$ is fixed, using a similar procedure as in Step 4(a)
```

In each iteration, we conduct these hypothesis tests for the proposed updates to our estimate of $\mathbf{x}$ and we accept the update only if ρ_t is less than Δ. Note that under our simplified notation, $\mathbf{x}$ is really $\mathbf{x}_F$ and we conduct the hypothesis tests only on this subset of components. If the hypothesis tests fail, we increase the sample size so as to increase the precision in the proposed updates and pass the hypothesis tests. We stop the algorithm once the hypothesis tests fail after we have reached the full sample size. In practice, we will not conduct the hypothesis test for every column (row) of $\mathbf{H}$ ($\mathbf{W}$), but randomly pick a representative sample J which we keep fixed during the course of the optimization. In our experiments we have always used $J = 10$. In theory, one should apply a correction for multiple hypothesis testing. For instance, the Bonferroni correction simply divides the significance level by the number of tests performed. These considerations are only important when one wants to use the hypothesis tests to determine a *principled* stopping criterion. In practice the significance level is often a tuning parameter to optimize the computational efficiency of the algorithm. The complete method, using BPP as a sub procedure, is summarized in Algorithm 2.

5 Experimental Results

We compared the performance of the original Block Principal Pivoting (BPP) algorithm to BPP using our Sub-Sampling scheme (BPP-SS) on 3 real world datasets: the AT&T dataset of faces[1], the MNIST handwritten digits dataset [2] and a dataset of HOG (Histogram of Oriented Gradients [4]) features from the INRIA pedestrian detection dataset [3].

Both algorithms were implemented in MATLAB. The experiments on the AT&T faces dataset were executed on a 2.2 GHz Core2 Duo, 3GB RAM machine running Windows 7 whereas the MNIST and HOG dataset experiments were run on a 2.93GHz 4 Intel(R) Xeon(R) X5570, 96GB RAM cluster node running Linux 2.6. For all datasets, we started out with a sample size around 500 and chose the threshold for failing a hypothesis tests to be 0.4. We always performed $J = 10$ tests at every iteration of the algorithm.

For BPP-SS, we stopped the algorithm when the hypothesis tests failed with the full dataset. We calculated the relative error or "residual" for BPP-SS as $\|\mathbf{A} - \mathbf{WH}\|_F / \|\mathbf{A}\|_F$ and used this as the stopping criterion for the BPP algorithm. We have verified that our stopping criterion resulted in very similar (in fact slightly smaller) residuals as the ones reported in [7]. We ran 10 trials on each dataset using the same initial values for $\mathbf{W}$ and $\mathbf{H}$ for both algorithms in any particular trial. We measured the average running time of both algorithms and the residual. For each trial (using the same initialization), we measured the speed-up factor as the ratio between the running time of BPP with respect to that of BPP-SS, and also its standard deviation. Note that because the algorithms may converge to different local minima for different trials, the standard deviation of the running times between trials is high and the only good comparison is the mean *paired* speed-up and its standard deviation. Since we are measuring the speed-up *ratio*, the mean and standard deviation refer to the geometric mean and geometric standard deviation respectively.

For tuning our parameters, our first experiments were on the AT&T database of faces which consists of 400 face images of 40 different people with 10 images per person. Each face image has 92×112 pixels and we obtained a 10304×400 matrix. The average performance of both algorithms are shown in Figure 3 for different values of k (the rank of $\mathbf{W}$ and $\mathbf{H}$).

k	Residual	Running Time (sec)		Speed Up	
		BPP	BPP-SS	Mean	Std. Dev.
16	0.1888	60.53	42.85	1.90	1.21
25	0.1725	106.46	90.57	2.18	1.30
36	0.1591	179.99	143.05	1.95	1.14
49	0.1475	560.64	467.63	2.25	1.26
64	0.1372	407.29	303.99	2.26	1.16
81	0.1284	707.81	459.14	2.28	1.20

Figure 3: Performance comparison on the AT&T face data set.

k	Residual	Running Time (sec)		Speed Up	
		BPP	BPP-SS	Mean	Std. Dev.
10	0.5936	79.2	54.41	1.21	1.43
15	0.5534	271.18	122.39	1.96	1.37
20	0.5228	303.92	150.23	1.96	1.37
25	0.4929	432.79	205.99	2.48	1.50
30	0.4673	578.28	241.86	2.79	1.32
35	0.4441	636.84	262.18	2.44	1.31
40	0.4225	742.26	285.81	2.82	1.30
45	0.4045	1021.1	430.76	2.79	1.54
50	0.3875	1409.9	620.75	2.49	1.41

Figure 4: Performance comparison on the MNIST database of handwritten digits.

We show the residuals over time for k =16 in Figure 5. Note that the residual can go up for BPP-SS when we increase the sample size, since we also introduce more NNLS sub-problems to be solved. We show the cumulative time taken as a function of iteration in Figure 8. The growth of sample size, i.e. the number of rows of $\mathbf{W}$ used to solve for $\mathbf{H}$, vs iteration is shown in Figure 6. In Figure 9, the time taken per iteration is shown. Note the spike in processing time, when the sample size is increased. In Figures 7 and 10, we show the basis learned using the BPP and BPP-SS algorithms respectively. Note, that the two algorithms may converge to different local minima even though they are initialized with the same values of $\mathbf{W}$ and $\mathbf{H}$. However, these solutions are equivalent in terms of residual error. Qualitatively, we have observed that the variability between the solutions produced by BPP and BPP-SS is no greater than the variability between solutions computed by BPP with different initializations.

Our next set of experiments was on the MNIST database of handwritten digits. We used 60000 images of handwritten digits, each 28×28 pixels, giving a 60000×784 matrix. Results are shown in Figure

[1]http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[2]http://yann.lecun.com/exdb/mnist/

[3]http://pascal.inrialpes.fr/data/human/

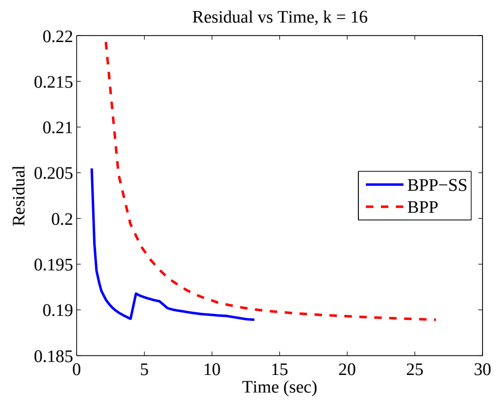


Figure 5: Residual vs Time, k = 16.

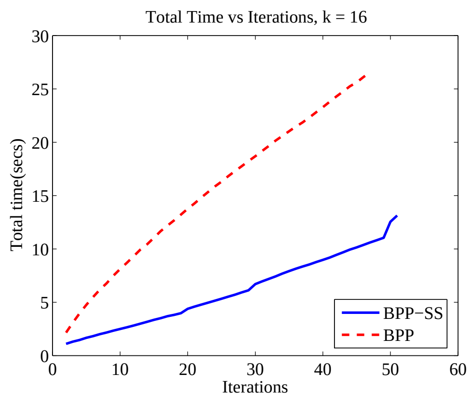


Figure 8: Total Time vs Iteration, k = 16.

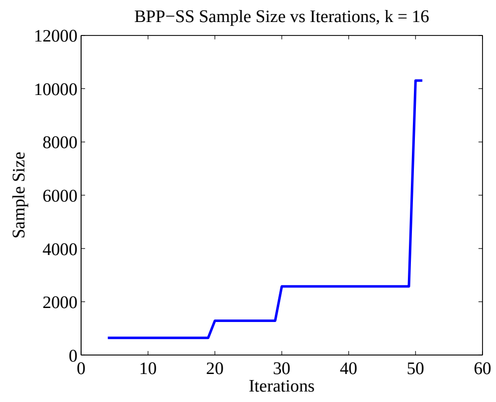


Figure 6: Sample Size vs Iteration, k = 16.

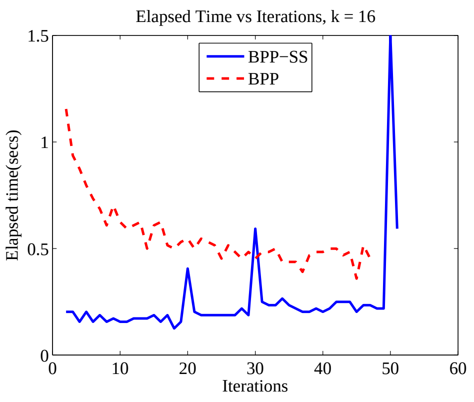


Figure 9: Elapsed Time per Iteration, k = 16.

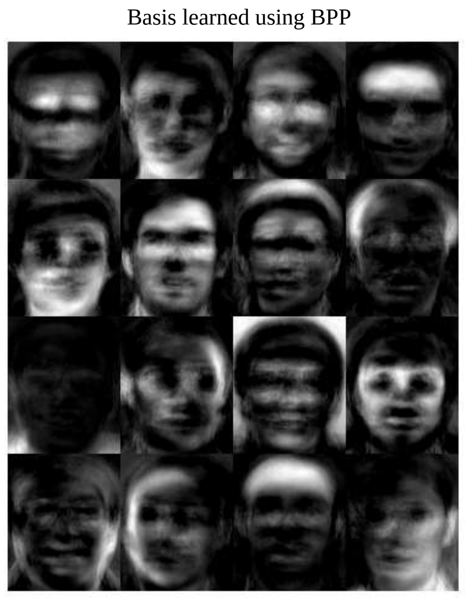


Figure 7: Basis learned using BPP, k = 16.

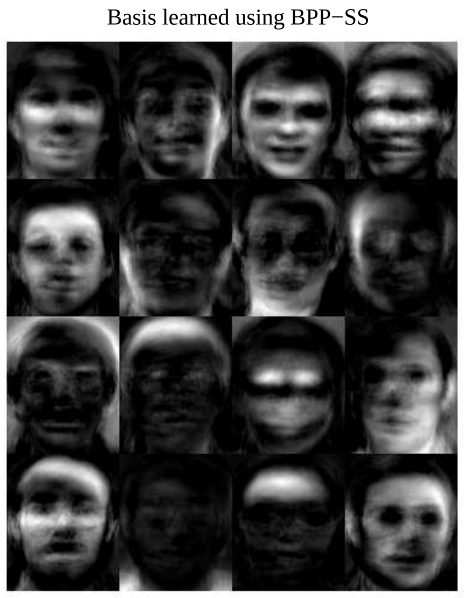


Figure 10: Basis learned using BPP-SS, k = 16.

k	Residual	Running Time (sec)		Speed Up	
		BPP	BPP-SS	Mean	Std. Dev.
5	0.3818	2609.9	943.21	2.06	1.79
10	0.3632	7858.4	2084.6	4.20	1.40
15	0.3489	14763	4049.6	3.09	1.46
20	0.3394	13508	5240.6	3.67	1.73
25	0.4894	14976	3929.6	5.39	1.54
30	0.3232	18127	7347.8	5.10	1.38

Figure 11: Performance comparison on the HOG features dataset.

4. Our final experiments were on a dataset consisting of HOG features of over 5 million windows from the INRIA pedestrian detection dataset. Each scanning window was divided into 14×6 cells of 8×8 pixels and each cell was represented by a gradient orientation histogram using 13 bins resulting in a 1092 element vector per window. We used only a 1 million subset of windows giving a $1,000,000 \times 1092$ matrix, because of memory constraints. This is an extremely large dataset which allowed the BPP-SS algorithm to achieve very significant speed-ups (Figure 11).

Our experiments on these three real world datasets clearly demonstrate that the proposed sub-sampling method significantly improves the NMF Block Principal Pivoting algorithm, especially on large datasets.

6 Conclusion

Building upon the BPP algorithm, we developed a very efficient algorithm for solving the NMF problem. In the initial iterations, the asymptotic normality of the least squares estimator allows us to work with a few samples of the data instead of the whole matrix, thereby saving a lot of computational time.

We advocated the view that statistical estimation problems are not merely mathematical optimization problems once the loss function has been decided on (see also [2] and [18] for a discussion of how optimization and learning can interact in interesting ways). A mere mathematical view ignores the important insight that the loss function is a random object that fluctuates under resampling the dataset. As such, statistical estimation problems have an intrinsic scale of precision that is larger than machine precision. Moreover, far from convergence, the precision necessary to update parameters is much smaller than close to convergence, opening the door for speeding up the learning process.

Our method crucially depend on the central limit theorem. We have indeed observed the method to fail when central limit tendencies are absent. This appears to be the case when the data-matrix is very sparse. We have also tried our method on tensor factorization problems. In this case the necessary reshaping operation of the tensor in each iteration induces complicated dependencies and non-Gaussian behavior (presumably due to multiplication of random variables).

However, despite the above exceptions, many interesting learning algorithms exist to which our ideas can be applied. As we have shown, our method is often orthogonal to speedup methods developed in the optimization literature. We are currently applying similar insights to improve the performance of SVM, logistic regression and LASSO algorithms, but we believe our ideas are much more widely applicable yet.

Acknowledgements

The authors thank Alexander Ihler and the anonymous reviewers for various helpful suggestions. This material is based upon work supported by the National Science Foundation under Grant No's. 0447903, 0535278, 0914783, 0928427, 1018433.

A Calculating the Rotation Matrix

Given two vectors $\mathbf{v}_1,\ \mathbf{v}_2 \in \mathbb{R}^{d\times 1}$ such that $\|\mathbf{v}_1\|_2 = \|\mathbf{v}_2\|_2$, we describe an efficient method [4] for calculating a rotation matrix $\mathbf{S}$ such that $\mathbf{S}\mathbf{v}_1 = \mathbf{v}_2$. First, consider the "thin" QR decomposition of the matrix $\mathbf{V} = [\mathbf{v}_1\mathbf{v}_2] = \mathbf{QR}$ where $\mathbf{Q} = [\mathbf{q}_1\mathbf{q}_2] \in \mathbb{R}^{d\times 2}$ is orthogonal and $\mathbf{R} = [r_{ij}] \in \mathbb{R}^{2\times 2}$ is upper triangular. Then, $\mathbf{q}_1$ and $\mathbf{q}_2$ form an orthonormal basis for $\mathcal{P}_V$, the sub-space (plane), spanned by $\mathbf{v}_1$ and $\mathbf{v}_2$.

Now, let us consider rotating $\mathbf{v}_1$ around an 'axis' orthogonal to $\mathcal{P}_V$. The projection of $\mathbf{v}_1$ onto $\mathcal{P}_V$ is $\mathbf{Q}\mathbf{Q}^T\mathbf{v}_1$ and the projection onto the axis is given by $(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{v}_1$. We only need to consider the rotation of the component projected onto $\mathcal{P}_V$, since the other component is invariant to the rotation. Therefore, let $\mathbf{U}$ be the 2×2 rotation matrix which aligns the projection of $\mathbf{v}_1$ onto $\mathcal{P}_V$ with the projection of $\mathbf{v}_2$ onto $\mathcal{P}_V$, i.e. $\mathbf{Q}\mathbf{Q}^T\mathbf{v}_2 = \mathbf{Q}\mathbf{U}\mathbf{Q}^T\mathbf{v}_1$. Adding on the components of $\mathbf{v}_1$ orthogonal to $\mathcal{P}_V$ to the rotated version of the projection of $\mathbf{v}_1$, we have $(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{v}_1 + \mathbf{Q}\mathbf{U}\mathbf{Q}^T\mathbf{v}_1 = (\mathbf{I} - \mathbf{Q}(\mathbf{I} - \mathbf{U})\mathbf{Q}^T)\mathbf{v}_1 = \mathbf{v}_2$. Thus, the d dimensional rotation matrix we seek is $\mathbf{S} = \mathbf{I} - \mathbf{Q}(\mathbf{I} - \mathbf{U})\mathbf{Q}^T$.

Since $\mathbf{v}_1 = r_{11}\mathbf{q}_1$ and $\mathbf{v}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2$, it is easy to see that:

$$\mathbf{U} = \frac{1}{\sqrt{r_{12}^2 + r_{22}^2}} \begin{bmatrix} r_{12} & -r_{22} \\ r_{22} & r_{12} \end{bmatrix}$$

Note that, for computing the rotation matrix in 8, $\mathbf{v}_1$ and $\mathbf{v}_2$ are related by $\mathbf{v}_2 = [\|\mathbf{v}_1\|_2; \mathbf{0}]$, and the above procedure is very efficient.

[4]http://forums.xkcd.com/viewtopic.php?f=17&t=29603

References

[1] D. P. Bertsekas. *Nonlinear programming.* Athena Scientific, Belmont, Mass, 1999.

[2] L. Bottou and O. Bousquet. Learning using large datasets. *Mining Massive DataSets for Security, NATO ASI Workshop Series*, 2008.

[3] L. Boyles. Statistical tests for optimization efficiency. Master's thesis, University of California, Irvine, 2010.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[5] D. Kim, S. Sra, and I.S. Dhillon. Fast newton-type methods for the least squares nonnegative matrix approximation problem. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[6] H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[7] J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*, pages 353–362, 2008.

[8] H.J. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications.* Springer Verlag, 2003.

[9] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Society for Industrial and Applied Mathematics, 1995.

[10] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998.

[11] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[12] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[13] K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. In *Biometrika*, volume 57, pages 519–530, 1970.

[14] Genevieve B. Orr. Removing noise in on-line search using adaptive batch sizes. In *NIPS*, pages 232–238, 1996.

[15] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[16] J.C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control.* John Wiley and Sons, 2003.

[17] M. Verbeek. *A Guide to Modern Econometrics.* John Wiley and Sons, 2000.

[18] B. Yu. Embracing statistical challenges in the information technology age. *Technometrics, American Statistical Association and the American Society for Quality*, 49:237–248, 2007.